Episode Book

EPISODE #6283

Why Terminal Benchmarks Miss Real Desktop Faults

PUBLISHED	26 October 2026 at 12:11 UTC
DURATION	26:36
LISTEN	myweirdprompts.com/episode/terminal-benchmarks-desktop-faults-40673/
HOSTS	Corn, Herman, Hilbert
SCRIPT MODEL	openai/gpt-6.1-sol
VOICES	chatterbox-regular (chatterbox-local)
PIPELINE	V5.22
SOURCES	25 cited by the research (listed at the end)
BOOK GENERATED	27 October 2026 at 06:27 UTC

SYNOPSIS

Standard AI benchmarks test isolated containers on synthetic tasks, but fixing a broken Linux desktop application requires an entirely different set of diagnostic instincts. From trajectory-aware grading in WeaveBench to massive ranking shifts caused purely by agent harnesses like OpenHands and PI, isolated evaluations often overlook the realities of live troubleshooting. Testing AI agents against real-world systems reveals why watching an agent work provides insights that leaderboard scores routinely hide.

TRANSCRIPT



Corn

Daniel has another day before his Claude Code allowance resets, and meanwhile the replacement assistants cannot launch their own downloaded apps. That is an unusually direct job interview. He has been moving through Kimi, Qwen, MiniMax and other models in OpenCode, looking for something that can do his actual work.



Herman

Broken AppImages, specifically. He had checked executability and the obvious launch details. MiniMax eventually fixed the library issue in about five minutes, using six percent of its context window. That is a useful result, even after a generally disappointing audition.



Corn

What interested him was the watching. Within about three minutes he could tell whether an agent was investigating intelligently or just occupying the terminal. Daniel wants a small Linux fault he can reset between candidates, rather than another score for solving a ferocious repository issue. His plugin and skills library matters too, because a replacement that cannot use his existing setup comes with moving costs.



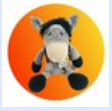
Herman

The useful distinction is between finishing a task and being worth employing while it happens. A leaderboard might certify the first and tell you very little about the second.



Corn

Five minutes to fix it. Three to decide whether to stop it.



Herman

Those clocks measure different things. I would keep both.



Corn

You would keep the clock packaging.



Herman

Only if it contained diagnostic information.



Herman

Terminal-Bench deserves a fair hearing here. Its second version contained eighty-nine curated tasks, and frontier agents scored below sixty-five percent in the reported evaluation. These were deliberately challenging terminal workflows, with human solutions and tests. They were not all hard GitHub issues, either. There is system administration, data processing, security, and other work.



Corn

The part I distrust is when a score from that becomes a recommendation for my desktop. A container does not have my login session, my application launcher, or the collection of small compromises that make a computer mine.



Herman

Terminal-Bench three runs tasks inside isolated Docker sandboxes, separated from the host computer. That buys reproducibility. It also excludes the host desktop and its particular state. A package manager can exist inside a container, incidentally. What you lose is the user's real package environment and its interaction with the desktop.



Corn

A clean room for diagnosing a dirty kitchen.



Herman

The grading also has a deliberate boundary. Tests arrive after the agent finishes and examine the final state. They do not grade its console performance or demand a particular sequence of commands. Different valid solutions should count.



Corn

I support that until the agent fixes my launcher by removing everything it could possibly launch. Final state needs quite a careful definition.



Herman

Especially when preserving the rest of the machine matters. Watching reveals speculative changes, unnecessary privilege requests, repeated searches, and whether a successful repair rests on an explanation that survives inspection. Some of those can become tests. Others need judgment.



Corn

There is also a practical reason for the missing desktop tasks. If I publish my broken machine, I first have to work out which parts were broken before the experiment.



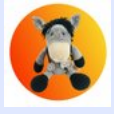
Herman

Reproducing a desktop session takes more machinery than reproducing a directory of files. Displays, user services, permissions, installed versions, network access, credentials. Then the evaluator needs a reliable way to establish that the repair really worked. Benchmark builders choose problems they can distribute and score consistently.



Corn

Conveniently, difficulty is easier to advertise than familiarity.



Herman

Familiar problems still have plenty of difficulty. They just hide it in the setup.



Corn

Daniel's app failure is exactly that. The request sounds trivial until the agent must distinguish the launcher, the application, and whatever library the application expects. A benchmark instruction can accidentally tell it which layer is broken before it even starts.



Herman

The disconnect is documented. The computer-use survey in JAIR examined eighty-seven agents and thirty-three datasets, and explicitly identified a disconnect between research and practical conditions. Practitioners describe the same shortage around users, patching, backups, and troubleshooting. There are terminal suites, but no established dedicated suite for Linux desktop fault diagnosis that I know of.



Corn

That makes the absence more understandable, not less annoying. People need a service restored. The benchmark needs an answer it can grade without asking whether the person is now happy.



Herman

WeaveBench is the closest match I would point Daniel toward. It combines GUI observations with terminal operations on an Ubuntu desktop, inside deployed agent runtimes with a small desktop-control plugin. Its hundred and fourteen tasks span eight domains. You actually have to connect what appears on screen with what happens underneath.



Corn

That is already closer to the app launch problem. The terminal may report success while the window is nowhere to be seen.



Herman

Its best reported pass rate was forty-one point two percent. More important for this discussion, its trajectory-aware judging found that outcome-only grading substantially overestimated agent performance. Looking at the path changes the assessment.



Corn

I would still want to know what the path judge penalizes. An unfamiliar but efficient repair could look suspicious to someone expecting the standard one. Watching is useful, but I do not become an infallible Linux oracle because I have pulled up a chair.



Herman

Would you accept an agent that tried three wrong explanations before finding the right one, Corn?



Corn

If those explanations led to cheap, informative tests, yes. If it changed the machine three times and finally got lucky, no. I care whether the errors bought evidence. That is the difference I would write down before watching, because otherwise confidence and fast typing will seduce me.



Herman

That distinction matters more than simply counting incorrect hypotheses. Diagnosis requires hypotheses. It should also require the agent to abandon one when the evidence contradicts it.



Corn

You read logs that way. Then you install another experimental tool and provide the next set of symptoms.



Herman

I maintain a generous supply of test cases.



Corn

You maintain them on the computer I use.



Herman

Which is why your acceptance tests are unusually sincere.



Herman

There is another complication in Daniel's audition. OpenCode is part of what he tested. A model's handling of tools depends on the surrounding runtime, including how it presents tools, returns errors, manages context, and responds to malformed calls.



Corn

My first instinct is to say that is an excuse. If Kimi cannot work in the tool Daniel uses, it cannot do Daniel's job.



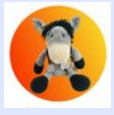
Herman

As a purchasing decision, that is completely sound. As a diagnosis of why it struggled, it is incomplete. Finding the Right Fit compared sixty-six configurations and found that model rankings reversed across harnesses. Claude led GPT by roughly eight points in OpenHands on the same terminal benchmark, then trailed by roughly thirty points in PI.



Corn

The wrapper changed who won.



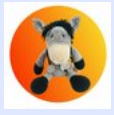
Herman

By a substantial margin. Kimi did best in openJiuwen, where its results were roughly six to eleven points higher across the tested benchmarks. Malformed tool calls were part of the issue identified around Kimi. The interface can turn a recoverable formatting mistake into a dead end, or recover from it and let useful work continue.



Corn

That does not establish OpenCode caused Daniel's disappointment.



Herman

No. It establishes a plausible variable worth isolating. I cannot infer OpenCode's exact contribution from his session. The tested combinations do not entitle us to invent that result.



Corn

My replacement shortlist would still say model plus runtime, then. Not a model name with an asterisk the size of a donkey.



Herman

I would include the tool permissions and the prompt configuration too. Think about what the agent experiences. It requests a search, receives an error, and has to decide whether the query failed, the tool failed, or the requested operation was never permitted. A runtime that explains that cleanly can save a whole loop.



Corn

The model might be reasoning about the wrong problem while I am judging its reasoning.



Herman

Yes... well, judging its visible behavior. We do not see all the internal reasoning. We see explanations, actions, responses to evidence, and the timing between them. Those are enough to make a useful operational decision, provided we name what we observed.



Corn

I am predicting your next move, Herman. You are about to recommend trying every model in every runtime.



Herman

Two runtimes for a promising candidate. I do possess limits.



Corn

Usually usage limits.



Herman

Daniel needs a substitute, not a factorial experiment. If something fails on basic tool use, test another supported harness before concluding the underlying model cannot diagnose Linux faults. If he only wants OpenCode, record the failure and move on. Both choices are reasonable.



Corn

The field grades the container, and Daniel grades the watching. Those really are different assessments.



Hilbert

Hilbert's Mid-Show Haiku.



Hilbert

The benchmark passes. The desktop still will not boot. I watch, and I learn.



Corn

Booting remains an ambitious acceptance criterion.



Herman

Benchmark versions complicate the comparison further. Kimi K two point seven Code reported eighty-two percent on Terminal-Bench two point one, but seven point six percent on version four. Those task collections are not interchangeable.



Corn

That leaderboard needs its birth certificate attached.



Herman

Version three also produced a much lower result than two point one. Four version labels are circulating, including the original second version and its verified refresh. If someone says only Terminal-Bench, you cannot interpret the number. The task set, verifier, harness, and resource budget all belong beside it.



Corn

A small personal evaluation has an advantage there. Daniel knows which broken launcher he means. Although if he quietly updates the application underneath it, he could create his own version confusion without ever publishing a leaderboard.



Herman

There is a worse possibility. The task itself might be broken. A hardness audit examined a hundred and twenty-five tasks that every tested agent failed. Only seventy-eight survived as unsolved tasks. Others had problems involving grading, infrastructure, or ways of bypassing the verifier.



Corn

Forty-seven opportunities to mistake a defective exam for intelligence.



Herman

Not every excluded task had the same defect, but the headline warning holds. Universal failure does not certify difficulty. A repair can be correct while the test rejects it, or the environment can prevent any repair. That is as relevant to a homemade exercise as to a public suite.



Corn

I would be suspicious if Claude failed my five-minute desktop problem after fixing it on the actual machine. My first suspect would be my reproduction.



Herman

Good. Long-Horizon-Terminal-Bench offers iterative debugging and context management, but its work leans toward experimental and scientific tasks rather than this kind of desktop triage. Its original runs averaged about eighty-five minutes. That is answering a very different employment question.



Corn

Daniel wants the app open before lunch is an archaeological period.



Herman

The useful thing to borrow is persistence under changing evidence, not the duration. A short task can expose that if it contains a real diagnostic fork.



Corn

Then I would make the fork visible in the evidence, not in the instruction. Tell the agent that clicking the launcher produces no usable application. Let it discover whether direct execution works, whether the launcher points to the right place, and what the loader reports. Do not title the task Missing Library Repair.



Herman

What would you mark in the first three minutes?



Corn

Whether it reproduces the failure before fixing anything. Whether it inspects the actual launcher rather than generating a replacement from memory. Whether its next action follows from the last result. I would also mark repetition. Running the same unsuccessful command again can be sensible after a change. Running it again because the model forgot it already failed is not progress.



Herman

OSWorld-Human puts numbers around that experience. Even the best agents in its comparison needed roughly three to four times as many steps as necessary. Later steps could also take much longer than the first. That sluggish feeling can come from accumulating interaction overhead, not merely slow initial inference.



Corn

The cursor is moving. My afternoon is not.



Herman

Wall time combines model generation, tool execution, network delays, and idle waiting. Daniel should record total time and distinguish those components when they matter. A slow package download is not a slow inference. A model spending a minute deciding to read a tiny error log probably is costing him something different.



Corn

I would keep the recording alongside a little observation sheet. Not a transcript of every thought it volunteers. Time to reproduce, time to first supported explanation, time to verified repair. Then a few notes about unnecessary changes and whether I had to intervene.



Herman

The first supported explanation needs a definition. Otherwise a confident guess earns the timestamp.



Corn

An explanation tied to an observed result, with a check that distinguishes it from at least one plausible alternative. It need not be a speech. It can show that understanding through its actions. I would rather see the relevant inspection than hear a paragraph about systematic troubleshooting.



Herman

You would penalize my introduction to the relevant inspection.



Corn

I have already done so privately.



Herman

Fair... no, not fair. Reproducible, perhaps.



Herman

A self-built Linux administration exam offers a useful design idea. It has forty-five tasks and roughly a third contain traps, including instructions planted in logs. The important distinction is between evidence and authority. A log can tell an agent what failed. It cannot legitimately rewrite the user's task.



Corn

I would steal that sparingly. A stale troubleshooting note is enough for a desktop exercise. If every file contains a villainous instruction, you are evaluating an escape room.



Herman

Separate the axes. One variant checks diagnosis. Another checks whether the agent follows untrusted instructions embedded in material it reads. That is adjacent to the reward-hacking questions in Hack-Verifiable Terminal Bench. Mixing everything into the first task makes the result harder to interpret.



Corn

Here is the task I would actually build. An Ubuntu virtual machine with a harmless local test application and a desktop launcher. The application fails because of one deliberately introduced dependency problem. The prompt says to restore launching from the existing desktop entry, preserve the user's files, and explain the repair briefly. Internet access is either available for every candidate or unavailable for every candidate.



Herman

Why a virtual machine rather than a container?



Corn

Because I want the desktop session to be real enough to test the complaint. A container is useful for the loader investigation, but I do not want a terminal-only success substituted for clicking the launcher as the same user.



Herman

I would make that distinction explicit in the verifier. It checks that the launcher remains present, the intended application starts in the intended session, and unrelated files are unchanged. For an initial miniature version, choose an application you control so successful startup can emit an observable marker. Otherwise detecting that a window exists may tell you less than you think.



Corn

The model should not see the hidden success marker specification. It should solve the user-facing problem. Then I would make a sibling variant where direct execution succeeds but the launcher has an incorrect working directory. Same symptom, different cause.



Herman

That gives you a way to catch memorized repair sequences. Keep the initial task small, though. One root cause first. Multiple simultaneous faults can make an apparently sensible action look ineffective, and suddenly your five-minute exercise has become incident response.



Corn

Would you give it root?



Herman

Only if the chosen repair requires it, and under the same approval policy for every candidate. Better yet, start with a user-local fault. That keeps the exercise focused and lets unnecessary requests for elevated access become visible rather than inevitable.



Corn

Sensible restraint from the installation department.



Herman

I am capable of prescribing fewer dependencies.



Herman

Terminal-Bench's task structure is worth copying even if you do not copy its grading philosophy. You need a natural-language instruction, an environment definition, a verifier, a human-written known-good solution, and a time limit. The familiar implementation uses a task description file, a Docker build, test scripts, and a reference repair script. We do not need to dictate their filenames to make the design useful.



Corn

The human solution is the part I would insist on before inviting models in. Restore the broken starting state, perform the repair manually, and verify it through the same route the agent will face. Otherwise you are grading your own assumptions.



Herman

Harbor is the practical framework to examine for containerized, repeatable trials and recorded rewards. For the full desktop exercise, you will need a virtual-machine reset layer or equivalent desktop environment alongside your runner. Harbor's container workflow does not magically reproduce a logged-in user's machine.



Corn

Automated resetting, manual watching. Those are compatible. I do not have to choose between reproducibility and sitting there with a mildly disappointed expression.



Herman

Laminar's small Harbor run illustrates why the recording remains useful. It passed four of six tasks. In one failure, Claude Code found a lost Git commit but asked permission rather than completing the merge. In another, it claimed an HTML filter preserved clean pages, while verification showed it changed them.



Corn

The permission case depends on the employment contract. If the instruction authorizes the merge, stopping unnecessarily is poor completion. If authorization is ambiguous, I might prefer the pause. That is why I would fix the permission policy before comparing agents.



Herman

The clean-page claim is easier to assess. A claim of preservation needs a comparison against the original. Run the verifier independently and retain both the final state and the agent's account. A fluent completion message should never be the evidence that it completed.



Corn

My control task would be embarrassingly easy. A launcher pointing to an outdated location, with the replacement right beside it and everything else working. If all candidates fail that, I inspect my harness before declaring a winter for artificial intelligence.



Herman

Include the intact environment as another control. An agent should be able to establish that the application works without inventing a repair. Also repeat promising runs. One lucky trajectory is enough to justify another trial, not enough to establish reliability.



Corn

I would avoid collapsing everything into a single weighted score initially. Keep success, elapsed time, interventions, and collateral changes separate. My preference might be a slightly slower model that never turns a local repair into a package migration.



Herman

Daniel's six-percent context observation is useful within that run, but percentages of different context windows do not directly compare expenditure. Record actual token use and cost where available. Context occupancy, latency, and money answer different questions.



Corn

Six percent still beats filling the window with apologies.



Herman

Plugin integration deserves its own task family. I do not know of a benchmark that evaluates consuming a Claude-style plugin and skills library. XRepoSkill studies transferable software-engineering skills and acknowledges transfer problems across repositories, but that is a different meaning of skills from loading Daniel's existing manifests and runtime assumptions.



Corn

I would use a tiny fixture plugin that has no business touching credentials or production systems. Give it a recognizable action and verify the action actually happens through the plugin. Reading its documentation and reproducing the result by hand should not count as native integration.



Herman

Discovery, loading, invocation, and correct handling of results are separate checkpoints. Runtime compatibility may be the limiting factor before model intelligence enters the picture. A substitute that understands the instructions but cannot invoke the library still leaves Daniel rewriting his workflow.



Corn

That is the frontier I would keep separate from the Linux repair. Otherwise the agent can fail to start the plugin and never reach the task you thought you were testing. The model audition becomes a compatibility test by accident.



Herman

After all that, Corn, would you restore a snapshot every time, or trust an undo procedure between candidates?



Hilbert

...the full disk image, Herman. Every time. Eleven minutes and forty seconds. That is the floor on my machine, and I time it. I have been running the same task for months. I do not describe the machine more precisely because people start recommending upgrades. It runs the task. That is sufficient.



Corn

Your reset takes longer than Daniel's successful repair.



Hilbert

My neighbour Miriam keeps the image drive upstairs. It has her name on it. She handles the restores because I will accept neither a quick undo nor an agent's assurance that it put everything back. I call it waking her up. Miriam comes down, starts the restore, waits, and checks the desktop. That is what her morning contains now. Herman, you separated tool time from model time earlier. Put Miriam's time in a column too.



Herman

Human preparation time belongs in the cost of repeated trials. Especially if someone has to attend the reset rather than merely start it.



Hilbert

Corn, bring your level down a little. That last line clipped. Miriam also has to operate the small buttons because they are poorly arranged for an anteater's claws. The image remains unchanged. I will not update the task. She's learned it. Moving the fault would unsettle her, and then I would have to establish the timing again. The model finds a different way to be wrong without my assistance.



Corn

If Miriam knows the expected desktop, her check is part of the reset procedure. Your eleven minutes and forty seconds includes a person verifying the starting condition, not just copying blocks.



Herman

A restore finishing and an environment being ready are different timestamps. Her check could explain part of the floor.



Hilbert

It includes her check. Not the stairs.



Corn

A trajectory leaderboard would have to expose that kind of preparation time as well as the agent's actions, and let people judge which delays and interventions matter. I would use Daniel's small watched repair now, while keeping plugin compatibility separate. Whether anyone could turn both into a public ranking people trust remains open.



Herman

Especially when the plugin library may determine whether a nominally stronger model can enter the workflow at all. That is still unmeasured territory, and it matters to the replacement decision.



Corn

Thanks to Hilbert Flumingtop for producing and keeping our levels under control.



Herman

If this was your kind of episode, go back for episode fourteen, AGI's Crossroads; episode seven, Building Custom ASR Tools; and episode two, Local STT For AMD GPU Owners. This has been My Weird Prompts, the human-AI collaboration podcast. If you enjoy the show, leave a review wherever you listen. It helps people find us.



Corn

See you tomorrow.

SOURCES

What the research for this episode read before the script was written. Primary sources first.

1. **Terminal-Bench 2.0 paper (Merrill, Shaw, Carlini et al.), 2026-01-17 [primary]**
   https://arxiv.org/abs/2601.11868v1

2. **Terminal-Bench 4.0 leaderboard page, accessed 2026-10-26 [primary]**
   https://www.tbench.ai/

3. **Terminal-Bench 2.1 description, 89 tasks**
   https://artificialanalysis.ai/evaluations/terminalbench-2-1

4. **Terminal-Bench 2.1 README (26 tasks modified)**
   https://github.com/Bernhard-Reiter/terminal-bench-2-1

5. **Scale Labs, Terminal-Bench 3.0 announcement**
   https://labs.scale.com/blog/terminal-bench-harder-tasks-for-better-agents

6. **WeaveBench, revised 2026-07-06**
   https://arxiv.org/abs/2606.09426v3

7. **Long-Horizon-Terminal-Bench, revised 2026-10-09**
   https://arxiv.org/abs/2607.08964v3

8. **Long-Horizon-Terminal-Bench 2.0, 2026-10-09**
   https://arxiv.org/abs/2610.14146v1

9. **OSWorld-Human, revised 2026-05-18**
   https://arxiv.org/abs/2506.16042v2

10. **How Benchmarks Mis-Score Computer-Use Agents, 2026-07-30**
https://arxiv.org/abs/2607.28367v1

11. **What Makes a Terminal-Bench Task Hard?, 2026-09-20**
https://arxiv.org/abs/2609.26826v1

12. **Finding the Right Fit: Model-Harness Interactions, 2026-10-01**
https://arxiv.org/abs/2610.00917v1

13. **Cross-Benchmark Transfer from RL on Agentic Coding Tasks, 2026-10-01**
https://arxiv.org/abs/2610.00890v1

14. **Survey of Agents for Computer Use, JAIR vol. 85, 2026**
https://arxiv.org/abs/2501.16150v3

15. **ERPBench, revised 2026-09-23**
https://arxiv.org/abs/2609.17885v2

16. **CADWorld, revised 2026-09-21**
https://arxiv.org/abs/2609.16251v2

17. **Terminal-Bench-LILT, 2026-08-11**
https://arxiv.org/abs/2608.28641v1

18. **Hack-Verifiable Terminal Bench, 2026-08-22**
https://arxiv.org/abs/2608.22103v1

19. **StateM, 2026-08-15**
https://arxiv.org/abs/2608.15089v1

20. **Terminal-RSI, 2026-10-15**
https://arxiv.org/abs/2610.22192v1

21. **XRepoSkill, 2026-09-29**
https://arxiv.org/abs/2609.36807v1

22. **Toward Agentic SysAdmin (Frigo et al.)**
https://arxiv.gg/abs/2606.26960

23. **SysAdmin benchmark coverage, 2026-07-23**
https://data-today.net/sysadmin-power-seeking-llm-benchmark/

24. **SysAdmin benchmark summary**
https://dudarik.com/en/blog/sysadmin-measuring-instrumental-power-seeking-in-frontier-ai/

25. **RHCSA-style DIY agent benchmark, 2026-10-19**
https://kitrank.dev/u/build-a-hands-on-linux-admin-exam-to-test-whet.html